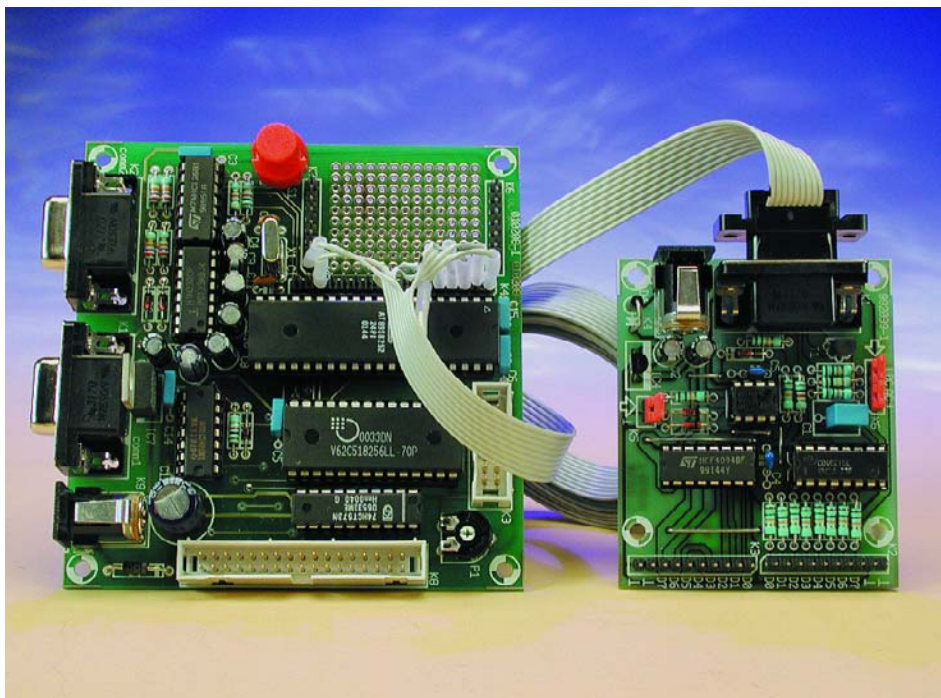


Port Line and ADC Extension for 89S8252 Flash Micro Board

using our Universal Interface for Windows

Design by B. Kainka

External components have to be used whenever the number of port lines on a microcontroller is insufficient. And when you need to process analogue signals instead of digital ones, it's time to look at adding an ADC.



If it is essential to keep the cost of a microcontroller extension circuit as low as possible, there's nothing to beat standard CMOS integrated circuits from the 4000 series. Only three port lines are required to drive a shift register that creates additional output lines: data, clock and a control line via an output

latch. The 'input' direction is equally simple because many A-D converters operate on the same principle of clocked serial data transmission.

The basis for extending the extremely popular **89S8252 Flash Micro Board** (December 2001) is the

Universal Interface for Windows described in the December 1999 *PC Topics Supplement*. The circuit diagram of the Universal Interface is shown in **Figure 1**. This unit is connected directly to the RS232 port and employs a type 4094 shift register for eight output lines, another shift register type 4021 for eight input lines and an inexpensive ADC type TLC549 to create an 8-bit analogue input with a measurement range up to +5 volts.

The extension circuit should be designed for minimum use of port line 'resources' because with RS232 there are only three output lines and four inputs available. Consequently, a common clock line is used for all three ICs, and they also share a chip select line. Only the data lines are available three times. When an extension circuit is hooked up to a microcontroller circuit, the same principle applies as when it is connected to a PC: the fewer lines are used up, the more ports remain available for other functions. With this in mind the author aimed at using the Universal Interface 'as is' with the 89S8252 Flash Micro Board.

The Universal Interface has a cou-

ple of zener diodes that serve to adapt the bipolar voltage swing on the RS232 lines to 0-5 V needed by the CMOS ICs. Level conversion is not required when a microcontroller is connected to the Universal Interface, and even has a small disadvantage in that fast pulse transients are 'rounded' to a certain extent by the stray diode capacitances. Fortunately, if you already have a built up Universal Interface, there is no need to remove the diodes as long as you do not run the microcontroller at maximum speed. Using moderate I/O and clock speeds will allow you to use one and the same Universal Interface both as a port line extender with your PC (connected via RS232) and your 89S8252 Flash Micro board. This may be very useful with experiments or educational use — depending on the assignment given to you, a proposed solution may be tested using PC software first and then evolve into a microcontroller project.

The Universal Interface is connected to Port1 of the controller, **not** to the Sub-D connector on the 89S8252 Flash Micro board. Port1 pin connections are available on pin-header K4 on the board. P3.5 is routed via K5, ground via K7 (see **Figure 2**). In addition to ground, a total of five signal lines are connected-up. Clock line 'Clk' and chip select line 'CS' are common to all three ICs. Next, there are individual data lines for digital output (Dout), digital input (Din), the A-D converter (Adin) and the counter input (Fin).

The extension is controlled in software using BASCOM-51. The modern BASIC offered by this compiler guarantees programs that are easily read and understood. If necessary, programs for other compilers or programming languages may be translated into BASCOM-51. Further advantages of BASCOM include direct control over port lines (impossible in BASIC-52) ready-made routines for an LCD and the possibility for assembler code to be embedded and so optimise your programs for speed.

Shift register 4094

Below we'll discuss the way the individual components are controlled. Each component is directly

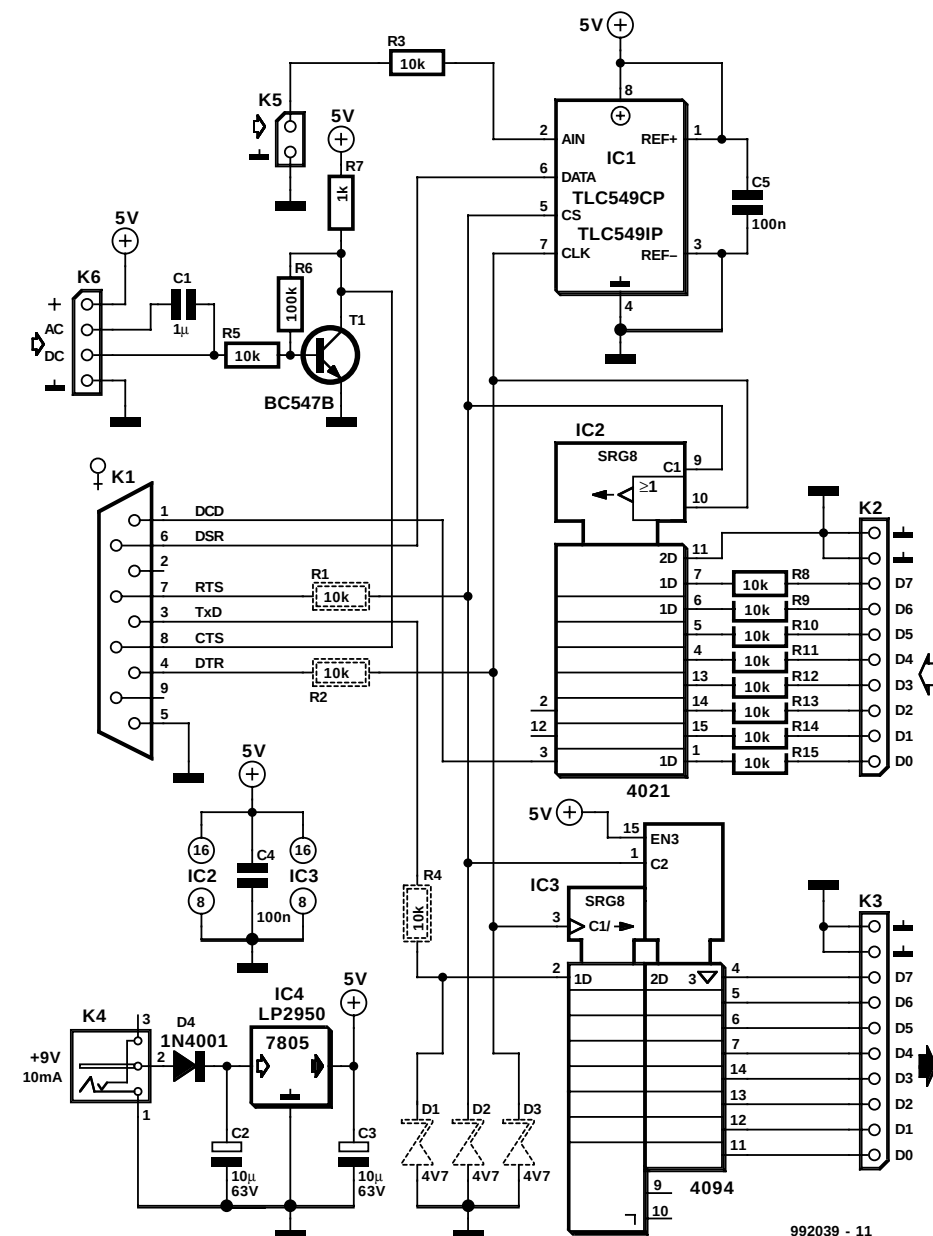


Figure 1. The Universal Interface for Windows may be connected to the 89S8252 Flash Micro board without modifications.

connected to the processor port and individually tested with the aid of a program. If you require just one component, you may want to fit it in the prototyping area on the Flash Micro board. We'll begin by tackling the 4094 shift register. **Figure 3** shows its connection to the processor port.

The actual control is not difficult. The processor supplies the first bit to dataline D and then generates a clock pulse at the CLK input. The databit is copied into the shift register on the positive edge of the CLK signal. The other seven bits are 'clocked in' in the same way. After

the last bit, a transfer pulse at the strobe input STR causes all data to be copied simultaneously into the output latch. Next, they appear as logic High and Low levels at the pins.

The datasheet of the 4094 will indicate that the order of the D0-D7 outputs has been reversed here. However, the names assigned to the output lines are arbitrary because we are free to begin with bit 0 or bit 7. Here, the bit order was used as found in the Windows software and the associated programming examples in Delphi [1] and Visual BASIC [2]. The advantage is that an experimental circuit may be tested using a Windows program before diving into the intricacies of microcontroller programming.

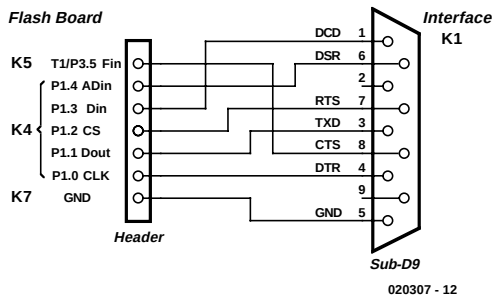


Figure 2. Cable link between Universal Interface and Flash Board.

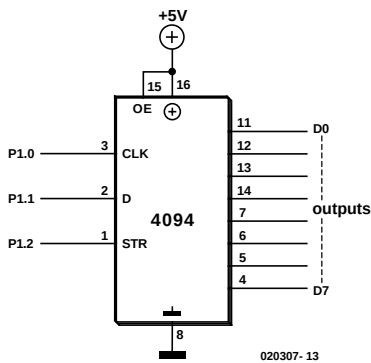


Figure 3. Connection of a 4094 shift register.

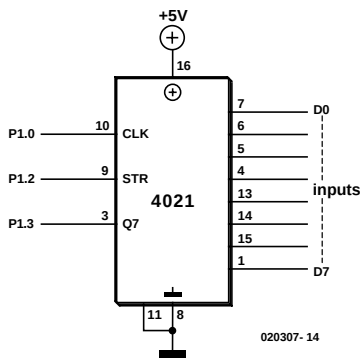


Figure 4. Creating eight inputs with the aid of a 4021.

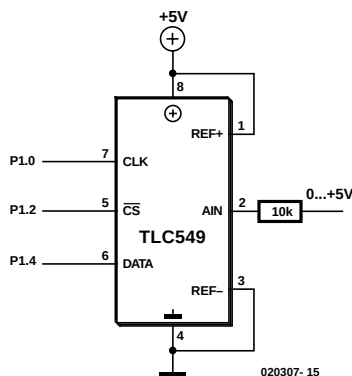


Figure 5. Connection of a TLC549 ADC.

The program in **Listing 1** was written to control the shift registers in BASCOM-51. The main program generates number sequences fed into the subroutine Out4094. A global variable called Outdat arranges the transfer.

Port lines P1.0, P1.1 and P1.2 are given names that reflect their function. This is done to keep the source code text legible. When using freely chose names, you may need to look carefully at the spelling to avoid confusion with reserved BASIC expressions — see, for example, 'Str/Strb'. The program editor will assist the user in this respect by highlighting all reserved words.

Data outputting employs a shift clock pulse in a loop with eight iterations. A variable Bitval copies bit 0 from the data transferred, and is then used to control the dataline. At the end of the loop the input data byte is shifted to the right by one bit. In this way, each successive bit in the word is placed onto the dataline, up to and including bit 7. Each successive Clock pulse is slightly delayed using the BASCOM-51 'Delay' command loaded with value 51 here for a delay of about 100 μ s. After the output loop, another 100- μ s long Strobe pulse arranges for the data to be copied into the output register.

You will now see repeating bit patterns at the shift register outputs. Each individual output operation takes 1.4 ms of which 0.9 ms can be traced back to the delays used here. When the circuit is connected directly to the controller, all delays may be removed resulting in a three-fold increase in outputting speed. The delays are however essential if the Universal Interface is applied.

Shift register 4021

Our next assignment is to control an input shift register type 4021 (**Figure 4**). As before, data inputs are labelled in accordance with the requirements of the Windows software. A strobe pulse at pin 9 causes parallel data to be copied in. Next, data bits are shifted at each clock pulse and appear sequentially at output Q7.

The BASCOM-51 program shown in **Listing 2** reads the extended port inputs and shows them on a 2-

line LCD. From version 2.0.10 onwards, the new compiler directive \$1cdrs = &H8002 ensures correct control of the Flash Micro Board. With older versions, it used to be difficult to actually employ the parallel LCD connection. The main program demonstrates the simplicity of dealing with LCD outputs. The top line shows the input states in decimal notation, the lower line, in hexadecimal notation. A wait command waitms 250 provides a sufficiently long delay and thus a legible readout.

The real read routine starts with a 100- μ s long Strobe pulse for the intake of parallel data. Next, using a loop each bit is read from the Din line and copied into bit 7 of the variable Indat. By means of a right-shift operation, the first bit finally lands at position 0, while the last bit remains at position 7, representing the value 128. After each bit a 100- μ s clock pulse moves the data by one position.

A/D converter TLC549

The TLC549 from Texas Instruments is a cheap and widely used A-D converters (**Figure 5**). At a resolution of 8 bits (256 steps), a voltage range of 0 to +5 V is divided into 20-mV increments. The TLC549 requires just two outputs and one input for its connection to a microcontroller. After a positive-going CS pulse of at least 20 μ s, eight databits may be clocked out of the IC. The transmission is similar to that of a shift register at a low level at CS. The shift clock is also the conversion clock for a successive approximation operation. Consequently, each read access produces the data of the last conversion.

The program shown in **Listing 3** illustrates the basic control method for the converter and the way data are sent to the LCD. The converter supplies data with values between 0 and 255, and multiplication by 20 yields the measured voltage in millivolts. This result is shown on the top line of the LCD. To make a 'volts' readout, a data type known as 'Real' would be required. Unfortunately that is not available in BASCOM-51, so as a recourse the lower line alternately shows the values before and after the decimal point. For example,

2300 mV is displayed as '2.300' V.

The A-D conversion routine Adt1c549 bears great resemblance to the reading of a 4021 shift register, although the order of the data is reversed — the higher value bit is read first, making it necessary for a shift-left command to be included in the read loop.

Comprehensive

As far as their control is concerned, the individual modules discussed above differ only marginally. Hence it makes sense to write a common subroutine. One clock signal is sufficient for both shift registers and the A-D converter. The chip select signal for the ADC doubles as the Strobe for the shift register, while the additional frequency input is read at the same time. Frequency measurements rely on Timer 1 of the controller. Input signals are conditioned by a small preamplifier as shown in **Figure 6**. Depending on the signal source, you can either use DC or AC coupling. The frequency meter created in this way has a range of up to about 400 kHz.

The program shown in **Listing 4** first initialises Timer 1 as a 16-bit counter with P3.5 as its external input. The actual measurement at a

gate time of 100 ms is carried out at the end of the universal hardware routine called Shiftall. Counter 1 is first reset to 0 and then started. After 100 ms, the counter is halted and read out. A resolution of 10 Hz is obtained by reading pulses within a gate of 100 ms. Multiplication by 10 then yields the frequency in Hertz.

The main program employs all inputs and outputs. The input voltage presented to the ADC, the digital input states and the current frequency of the signal applied to the counter input are all displayed. Furthermore, the results of the ADC are copied to the digital outputs and may be 'viewed' there by means of a set of LEDs.

The subroutine Shiftall caters for the common driving of all peripheral modules. The shared clock signal can make do without a dedicated delay routine because program parts that are necessary in any case provide the necessary delays of a few microseconds. The clocked data transmission takes less than a millisecond. This allows time-critical tasks to be handled, although it must be said that the frequency measurement has an additional time requirement of 100 ms.

(020307-1)

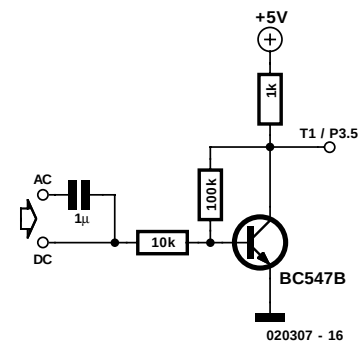


Figure 6. A small preamplifier for the frequency meter.

Note:

The BASCOM-51 programs discussed in this article are available from the Free Downloads page at www.elektor-electronics.co.uk. Item number **020307-1 I**, under month of publication.

For further reading

PC Interfaces under Windows, Elektor Electronics (Publishing), ISBN 0 905705 65 1.

Listing 1.

Output via a 4094 shift register

```
'-----
'      Shift4094.BAS
'      Elektor 89S8252 board
'-----
$regfile = "89s8252.dat"
Dim N As Byte
Dim Outdat As Byte
Declare Sub Out4094

While 1 = 1
  For N = 0 To 255
```

```
    Outdat = N
    Out4094
    Next N
  Wend
End

Sub Out4094
  Clk Alias P1.0
  Dat Alias P1.1
  Strb Alias P1.2
  Dim Count As Byte
  Dim Bitval As Byte
  Strb = 0
  Clk = 0
```

```
  For Count = 1 To 8
    Bitval = Outdat And 1
    If Bitval > 0 Then Dat = 1
  Else Dat = 0
    Clk = 1
    Delay
    Clk = 0
    Shift Outdat , Right , 1
  Next Count
  Strb = 1
  Delay
  Strb = 0
End Sub
```

Listing 2.

Data input via a 4021 shift register.

```
'-----
'      Shift4021.BAS
'      Elektor 89S8252 board
'-----
$regfile = "89s8252.dat"
Dim N As Byte
Dim Indat As Byte
Declare Sub In4021
```

```
Config Lcd = 16 * 2
$lcd = &H8000
$lcdrs = &H8002

Cls
While 1 = 1
  In4021
  Locate 1 , 1

  Lcd "Inputs=" ; Indat ; "
  Lowerline
```

```
Lcd "hex "
Lcdhex Indat
Waitms 250
Wend
End

Sub In4021
  Clk Alias P1.0
  Strb Alias P1.2
  Din Alias P1.3
  Dim Count As Byte
  Strb = 0
```

```

Clk = 0
Strb = 1
Delay
Strb = 0
Indat = 0

For Count = 1 To 8
    Shift Indat , Right , 1
    If Din > 0 Then Indat = Indat
    + 128
    Clk = 1
Delay
Clk = 0
Next Count
End Sub

```

Listing 3. Reading an ADC type TLC549.

```

'-----
'      Adtlc549.BAS
'      Elektor 89S8252 board
'-----

$regfile = "89s8252.dat"
Dim N As Byte
Dim Addat As Byte
Dim Voltage As Integer
Dim Millivolts As Integer
Dim Volts As Integer
Declare Sub Adtlc549

Config Lcd = 16 * 2
$lcd = &H8000
$lcdrs = &H8002

Cls
While 1 = 1
    Adtlc549
    Voltage = Addat * 20
    Locate 1 , 1
    Lcd "AD= " ; Voltage ; " mV "
    Lowerline

    Volts = Voltage / 1000
    Millivolts = Voltage Mod 1000
    Lcd "AD= " ; Volts ; "." ; Millivolts ; " V "
    Waitms 250
Wend
End

Sub Adtlc549
    Clk Alias P1.0
    Cs Alias P1.2
    Adin Alias P1.4
    Dim Count As Byte
    Addat = 0
    Cs = 0
    Clk = 0
    Cs = 1
    Delay
    Cs = 0
    For Count = 1 To 8
        Shift Addat , Left , 1
        If Adin > 0 Then Addat = Addat + 1
        Clk = 1
        Delay
        Clk = 0
    Next Count
End Sub

```

Listing 4. A suggested program to control it all.

```

'-----
'      Interface.BAS
'      Elektor 89S8252 board
'-----

$regfile = "89s8252.dat"
Dim N As Byte
Dim Outdat As Byte
Dim Indat As Byte
Dim Addat As Byte
Dim Voltage As Integer
Dim Freq As Long
Declare Sub Shiftall

Config Lcd = 16 * 2
Config Timer1 = Counter , Gate = Internal , Mode = 1

$lcd = &H8000
$lcdrs = &H8002

Cls
While 1 = 1
    Shiftall
    Outdat = Addat
    Voltage = Addat * 20
    Locate 1 , 1
    Lcd Voltage ; " mV "
    Locate 1 , 12
    Lcdhex Indat ; " "
    Lowerline
    Lcd Freq ; " Hz "
    Waitms 250
Wend
End

Sub Shiftall
    Clk Alias P1.0
    Dout Alias P1.1
    Cs Alias P1.2
    Din Alias P1.3
    Adin Alias P1.4
    Dim Count As Byte
    Dim Bitval As Byte
    Clk = 0
    Indat = 0
    Addat = 0
    Cs = 1
    Delay
    Cs = 0
    For Count = 1 To 8
        Shift Addat , Left , 1
        If Adin > 0 Then Addat = Addat + 1
        Shift Indat , Right , 1
        If Din > 0 Then Indat = Indat + 128
        Bitval = Outdat And 1
        If Bitval > 0 Then Dout = 1 Else Dout = 0
        Clk = 1
        Shift Outdat , Right , 1
        Clk = 0
    Next Count
    Cs = 1
    Delay
    Cs = 0
    T11 = 0
    Th1 = 0
    Start Counter1
    Waitms 100
    Stop Counter1
    Freq = Th1 * 256
    Freq = Freq + T11
    Freq = Freq * 10
End Sub

```